# Residential floor plan recognition and reconstruction

Xiaolei Lv, Shengchu Zhao, Xinyang Yu, Binqiang Zhao
Alibaba Group
{yanjun.lxl, shengchu.sc, xinyang.yu, binqiang.zhao}@alibaba-inc.com

## Abstract

*Recognition and reconstruction of residential floor plan drawings are important and challenging in design, decoration, and architectural remodeling fields. An automatic framework is provided that accurately recognizes the structure, type, and size of the room, and outputs vectorized 3D reconstruction results. Deep segmentation and detection neural networks are utilized to extract room structural information. Key points detection network and cluster analysis are utilized to calculate scales of rooms. The vectorization of room information is processed through an iterative optimization-based method. The system significantly increases accuracy and generalization ability, compared with existing methods. It outperforms other systems in floor plan segmentation and vectorization process, especially inclined wall detection.*

## 1. Introduction

Architectural floor plans, scaled drawings of apartments and building spaces, are utilized to assist users to decorate rooms, design furniture layout and remodel indoor spaces [14, 22]. Floor plan images, created by specific professional designers and architects, are rendered from vector-graphics representation, generated by AutoCAD [1], Sketchup [7] and HomeStyler [3]. However, after the above rasterization process, designers cannot modify the structure of the room and redesign flexibly. Therefore, accurately recovering vectorized information from pixel images becomes an urgent problem to be solved.

The above-mentioned problems have existed for decades. The generalization ability of traditional methods is weak, the whole process is difficult to automate, and a large amount of manual participation is required. The recognition of floor plans needs to consider various information as shown in Figure 1, such as room structure, type, symbols, text and scale. In recent years, with the rapid development of deep learning technology, related methods have made great progress in generalization. However, it is still a challenging problem to organically integrate various information and perform accurate vector reconstruction of the actual physical size.

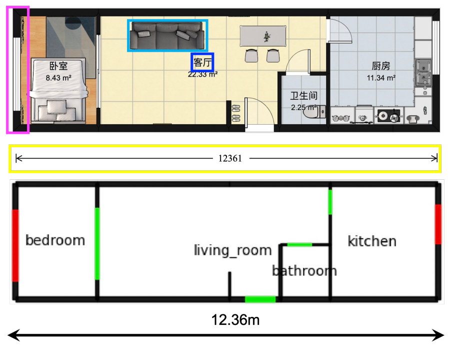

Figure 1. Pixel (top) and vector (bottom) floor plan image. The pixel floor plan image (top) contains various information, such as room structural elements (magenta), text (blue), symbols (cyan), scales (yellow). The original image comes from HomeStyler [3].

A novel floor plan recognition and reconstruction system, based on deep learning, is proposed, which combines the multi-modal information of floor plan images, such as room structure, type, symbols, text and scale. The system significantly outperforms existing methods, as shown in Table 1. Furthermore, we propose a dataset containing 7000 annotated images of residential apartments.

Our approach to residential floor plan recognition and reconstruction includes a number of technical contributions:

- An automatic framework for residential floor plan recognition and reconstruction that accurately recognizes the structure, type, and size of the room, and outputs vectorized 3D reconstruction results.
- An iterative optimization-based vectorization method for floor plan images that can accurately extract horizontal, vertical and inclined walls.
- A systematic method for recognizing the scale of floor plan images that can assist the 3D reconstruction of real physical size.
- The largest residential floor plan image dataset with

detailed vectorized and pixel annotations.

## 2. Related work

Floor plan drawings are crucial in real estate and design area. Systems of analyzing floor plans have existed for decades. Traditional methods [8, 9, 10] focus on directly processing low-level features. These systems produce a large number of hand-designed features and models. Ahmed *et al*. [10] introduce the idea of separation of text and graphics to analyze floor plan. Erosion, dilation and connected components detection are utilized to remove noise and extract text features. Wall information is extracted according to the size of connected components, vectorized methods, Hough transform and morphological approaches [8, 13, 25]. Speeded Up Robust Feature (SURF) is utilized to spot symbols [9], for instance, doors. Those systems mentioned above bring the problem of insufficient generalization ability. Thresholds and features are adjusted frequently by handcrafted operations instead of automatic methods.

With the development of deep learning techniques, the method of obtaining room structure has made significant process in generalization. Convolutional Neural Network (CNN) can create and extract advanced features to enhance the recognition performance of room elements [22, 29, 33]. Liu *et al*. [22] illustrate junctions of floor plans, for example, corners of walls, could be recognized by CNN. Combined with integer programming, a vectorized output is achieved. However, the approach has limitations, for instance, it is not able to detect inclined walls. Zeng *et al*. [33] improve performance of system by improving the loss function and adjusting the architecture of the deep neural network. Especially, room-boundary guided attention mechanism is developed and utilized to enhance the performance of pixel classification of floor plan images. Zhang *et al*. [34] adopt direction-aware kernels and Generative Adversarial Networks (GAN) to improve the efficiency and accuracy of image segmentation tasks. The disadvantage of [33] and [34] is that those methods do not obtain vectorized results that users can utilize. Also, text and symbols information are not detected and utilized sufficiently.

Methods for extracting text and symbolic information have developed significantly these years. Traditional methods [8, 10, 9] spend a huge amount of resources to eliminate noise before detecting symbols and text. However, CNN proves that it is able to extract important features automatically. Generally, Faster R-CNN [27] is used to detect sofa, closestool and other symbols [14, 29]. The disadvantage is that it is difficult to detect small objects, which leads to a decrease in network performance. Text is recognized by optical character recognition (OCR) frameworks, such as EAST [35] and CTPN [31]. Those OCR frameworks are composed of LSTM [15] network and CNN, bringing complexity to system.

Scale is a crucial part of a floor plan. Dodge *et al*. [14] illustrate a method to recognize the area of a room through text recognition. Combining with image segmentation results, scale could be calculated. This process may cause additional errors due to incorrect pixel classification.

A few datasets are available for recognition of floor plan images. Cubi-Casa5K [17], a dataset contains 5000 floor plan images of Finland, is wildly used to train and evaluate floor plan systems. However, the style is different from Asian regions hugely. Rent3d (R3D) dataset [21] collects 215 floor plans drawing for researchers. Liu *et al*. [22] mention they annotate 870 floor plan images manually to form a dataset, named R2V. The number of images in dataset R3D and R2V is far less than ours dataset.

## 3. RFP dataset

We have crawled 7,000 **R**esidential **F**loor **P**lan (**RFP**) data from Internet search engines, which are mainly home floor plans of Chinese urban buildings. We manually marked the starting endpoints, ending endpoints and thickness of walls. Each part of the wall is represented by a line with a width. We also marked the starting points, ending points and thickness of doors, windows and doorways. Doors, windows and doorways are all located on specific walls. Each room is surrounded by multiple parts of walls, and we manually marked the room type. Room types include living room, bedroom, bathroom, balcony, kitchen, library and others. We use rasterization to convert current annotations into pixel-by-pixel annotations, where each pixel represents a type. Please refer to the supplementary material for detailed annotations and statistics of the RFP dataset.

In the RFP dataset, 5,600 pictures were randomly selected as the training set, and the remaining 1,400 pictures were used as the test set. In addition, we selected 1,000 pictures with a text description of the room type and marked the room type. 1,000 pictures with furniture tags were selected for manual annotation for the training and evaluation of the symbol recognition model. 1,000 images with scales were labeled to train and verify the scale calculation module.

## 4. Method

Figure 2 is the schematic diagram of the entire recognition and reconstruction system. Our system is divided into two parts: recognition and reconstruction. The recognition part is responsible for detecting the area of the floor plan (Section 4.1), and identifying structural elements of the building (Section 4.2), auxiliary text and symbols (Section 4.3 ), and related scale information (Section 4.4) from image pixels. The reconstruction part includes converting the information obtained above into a vectorized expression

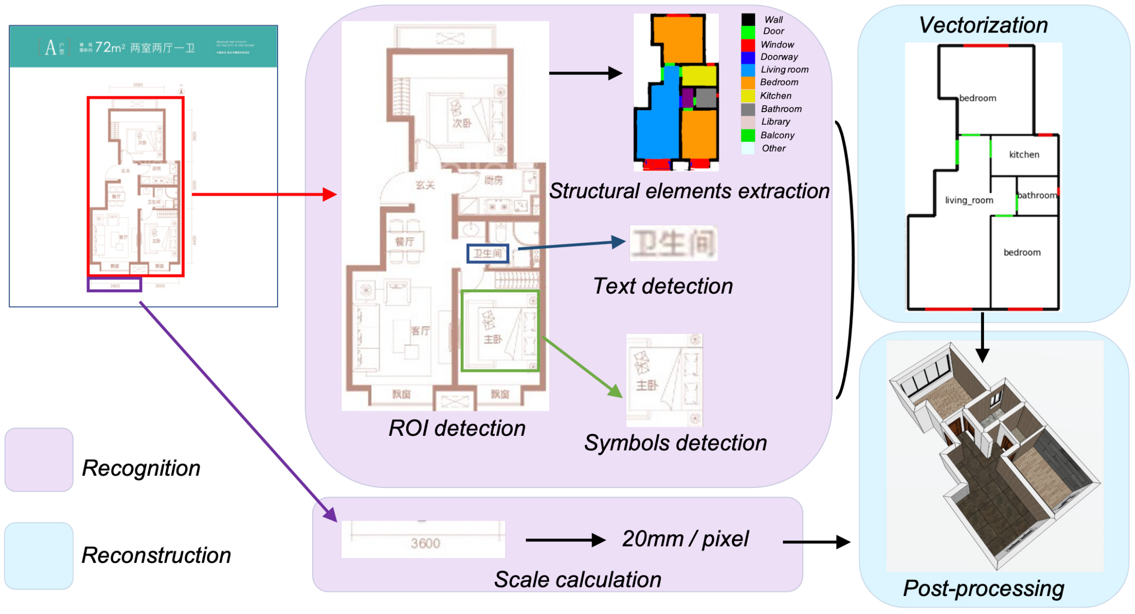

Figure 2. System overview, which combines the multi-modal information of the floor plan, such as room structure, text, symbols and scale. The original image comes from Lianjia [5]

(Section 4.5), and finally using a rule-based post-processing method (Section 4.6) to determine the type of doors and windows.

### 4.1. ROI detection

We observe that the effective floor plan area in the input picture may only account for <50% of the area. Please refer to the attachment for statistics on the percentage of effective floor plan area. Thereby, a detection network is utilized to obtain a compact floor plan area, which is different from previous related work [22, 33]. Most of the state-of-the-art object detectors utilize deep learning networks. Considering the detection efficiency and accuracy, YOLOv4 [11] is used as our basic detection model of region of interest (ROI) detection module. It is a lightweight and widely used deep neural architecture. The floor plan areas are detected by ROI detection module, then cropped as the input of the next modules.

### 4.2. Structural elements extraction

Similar to previous articles [33, 34], various kinds of structural floor plan elements (walls, doors, windows, doorways, *etc*.), especially for nonrectangular shapes and walls of nonuniform thickness, are expected to be recognized, as well as rooms types in floor plans, *e.g*. living room, bedroom, bathroom. Furthermore, the endpoints of doors, windows and doorways are expected to be identified accurately.

The DeepLabv3+ [12] is adopted as our basic network. The aspect ratio of ROI of images should be maintained during resizing process. The input should be resized to $w \times h$. We set $w = h = 512$. The system takes a color image of size $w \times h \times 3$ as input and outputs a feature map of size $w \times h \times (C + K)$. The first $C$ dimensions ($C$ = 12 for background, wall, door, window, doorway, living room, bedroom, kitchen, bathroom, library, balcony and other room) represent the prediction probability in semantic segmentation; the last $K$ dimensions ($K = 3$ for door, window and doorway) represent the heatmap in endpoint regression. Cross-entropy loss is frequently used in semantic segmentation tasks, which lacks spatial discrimination ability to distinguish between similar or mixed pixels [18]. We introduce affinity field loss [18] to incorporate structural reasoning into semantic segmentation. At the same time, we use opening regression loss to regress the boundaries of different elements (doors, windows, doorways). Finally, a method, using a network to learn weights between different tasks, is utilized [20]. It avoids those complex and time-consuming manual adjustment steps. Specifically, those weights are implicitly learned for each task through the homoscedastic uncertainty term.

- *Cross entropy loss:* $\mathcal{L}_{\text{ce}} = \sum_i \sum_c^C -y_i \log p_i(c)$ where $y_i$ is the label of the $i$-th floor plan element and $p_i(c)$ is the prediction probability of $i$-th pixel in category $c$.

- *Affinity field loss:* If pixel $i$ and its neighbor $j$ have the same categorical label, a grouping force is imposed to encourage network predictions at $i$ and $j$ to be similar. Otherwise, a separating force pushes their label predictions apart.

$$\mathcal{L}_{\text{group}}^{ic} = \sum_{j \in \mathcal{N}(i)} (1 - w_{ij}) D_{\text{KL}}(p_i(c) || p_j(c)) \quad (1)$$

$$\mathcal{L}_{\text{separate}}^{ic} = \sum_{j \in \mathcal{N}(i)} w_{ij} \max\{0, m - D_{\text{KL}}(p_i(c) || p_j(c))\} \quad (2)$$

$$w_{ij} = \begin{cases} 1 & \text{if } y_i \neq y_j \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$\mathcal{L}_{\text{affinity}} = \sum_i \sum_c (\mathcal{L}_{\text{group}}^{ic} + \mathcal{L}_{\text{separate}}^{ic}) \quad (4)$$

where $\mathcal{N}(\cdot)$ denotes the neighborhood. $D_{\text{KL}}$ is the Kullback-Leibler divergence between two distributions. The overall loss is the sum of $\mathcal{L}_{\text{group}}^{ic}$ and $\mathcal{L}_{\text{separate}}^{ic}$ over all categories and pixels.

- *Opening regression loss:* The set $S = (S_1, S_2, ..., S_K)$ has $K$ heatmaps, one per category, where $S_k \in \mathbb{R}^{w \times h}$.

$$\mathcal{L}_{\text{regression}} = \sum_q \sum_k^K ||S_k(q) - S_k^*(q)||_2^2 \quad (5)$$

Among them, $S_k^*(q)$ and $S_k(q)$ are the ground truth heatmap and predicted heatmap of location $q \in \mathbb{R}^2$ in the category $k$, respectively. We generate individual heatmap $S_k^*(q)$ for each category k. Let $x_{i,k} \in \mathbb{R}^2$ be the ground truth position of $i$-th opening endpoint of category $k$. The value of location $p \in \mathbb{R}^2$ in $S_k^*(q)$ is defined as

$$S_k^*(q) = \max_i \exp(-\frac{||q - x_{i,k}||_2^2}{\sigma^2}) \quad (6)$$

where $\sigma$ controls the spread of the peak. In order to distinguish adjacent peaks, we use the maximum value (rather than the average) of a single heatmap as the ground truth heatmap.

- *Multi task loss:* We add automatic weight adjustments for each loss and rewrite them, as explained in the supplementary material. The final loss is the sum of all losses.

$$\mathcal{L} = \mathcal{L}_{\text{ce}} + \mathcal{L}_{\text{affinity}} + \mathcal{L}_{\text{regression}} \quad (7)$$

### 4.3. Text and symbols detection

The text could provide extra information to refine room types. The text features of the same room area are the same between different images. Our system is simplified to a text detection model, instead of a redundant OCR framework. Small objects detection is very challenging. To this end, individual characters are combined into words and phrases to expand the size of an object so that it can contain additional semantic and contextual information [30]. YOLOv4 [11] is our basic network.

Symbols detection, as same as text detection, is utilized to revise types of rooms, for instance, closestool would help confirm the type of bathroom. Similar to the text detection above, the combination of symbols can expand the context information to help improve the detection performance. For instance, dinner table and chairs could be regarded as dinner table combo. The basic network is still YOLOv4 [11], while different data augmentation strategies need to be designed. Using Mosaic [11] makes networks concentrate on each part of the entire object, thereby greatly improving the generalization ability of the model.

### 4.4. Scale calculation

When it comes to 3D reconstruction of floor plans, scale information plays an important role. Numbers and lines are the basic elements of scale calculation system. The method focuses on four parts: line segments detection, numbers recognition, lines and numbers matching and scale calculation. The pipeline is shown in the supplementary material.
**Line segments detection.** In scale calculation, line segments detection aims at obtaining the pixel length of line segments by detecting endpoints. Endpoints are classified into 4 classes (up, bottom, left and right), according to the location in floor plan drawings. We formalize the line segment detection problem as an endpoint heatmap regression problem, which is similar to the opening regression loss in Section 4.2. The network is designed to a modified FCN network [24, 32], and backbone is Resnet50 [16]. The detailed architecture is shown in the supplementary material.
**Numbers recognition.** The digital recognition module is composed of number area detection, digital recognition and digital quantity regression. According to the direction of number areas, objects are divided into 3 categories: the horizontal direction, rotating 90 degrees counterclockwise to the horizontal direction, and rotating 270 degrees counterclockwise to the horizontal direction. Number areas detection is achieved by modified YOLOv4 [11]. Then, digits are also recognized by YOLOv4 [11]. The digits quantity result is calculated according to outputs of digits recognition. The other quantity of digits is regressed by modified VGG16 [28]. The number of areas will only be calculated, when those two results of quantity of digits are the same.
**Matching and calculation.** The line segment and number

region need to be matched. The distance between "line segment" and "number region" is described as the Euclidean distance between midpoints of the line segment and centers points of the number area. Scores of each pair is expressed as the reciprocal of its distance. A bipartite graph can be constructed through the relationship of lines and numbers. The line segments, and number regions are divided into two vertex sets, which are connected by edges with their scores. Then, the matching problem is translated to the maximum weight matching of a bipartite graph, which is solved by Kuhn-Munkres method. After that, matched pairs are generated, then scales are calculated. K-means is applied to those scales, then the largest quantity of class is chosen. Final output is averaged scale of the class.

## 4.5. Vectorization

In this section, we expect to convert raster information (structural elements, text and symbols) into vector graphics. The segmented floor plan elements can be divided into two categories [33], one is room-boundary pixels, and the other is room-type pixels. The room-boundary pixels represent walls, doors, windows and doorways, while room-type pixels are living room, bathroom, bedroom, *etc*. The core idea is to obtain vectorized contour information of room-type pixels for each room, then calculate the center line of the wall based on the contour information and room-boundary pixels. The room type can be refined by the text and symbol detection results. The position information of the doors, windows and doorways can be obtained according to the segmentation and regression results. Detailed steps are shown in Algorithm 1.

**Algorithm 1** Algorithm for vectorization

```
 1: Initialization;
 2: /* Room contour optimization */;
 3: for each room region $\mathcal{C} \in$ connected components do
 4:     while (condition a)$or$(condition b) do
 5:         Polygon vertex coordinate optimization;
 6:         Polygon vertex number reduction;
 7:     end while
 8: end for
 9: /* Center line optimization */;
10: while (condition x)$or$(condition y) do
11:     Wall junction coordinate optimization;
12:     Wall junction number reduction;
13: end while
14: Room type refine;
15: Opening endpoints extraction;
```

**Initialization.** Based on the results of pixel segmentation, a general method is utilized to determine the pixel width of the wall to assist the following calculations. Please refer to the supplementary material for the specific calculation method.

**Room contour optimization.** The uncertainty of segmentation results of objects at the boundary is relatively high [19], so the boundary has a complicated visual contour shape. In the residential floor plan image, most of the walls are horizontal and vertical due to aesthetic and architectural reasons. (If there is an inclined wall, the inclined wall is usually longer in length.) Therefore, for each connected area (*e.g.* a certain room), a simplified polygon is utilized to represent the contour. Initially, Douglas-Peucker algorithm [26] is utilized to get an initial polygon contour, then an iterative method is adopted to obtain simplified polygons from coarse to fine.

- *Room contour vertex coordinate optimization:* The polygon $\mathcal{P}$ has $N$ vertices and $N$ edges. $p_i \in \mathbb{R}^2$ represents the $i$-th vertex, $p_ip_{i+1}$ represents the $i$-th line segment, and $\overrightarrow{p_ip_{i+1}}$ represents the vector from $p_i$ to $p_{i+1}$. The pixel contour of the room region $\mathcal{C}$ is $\mathcal{B}$, with contour vertex $b \in \mathbb{R}^2$.

$$\mathcal{L}_{\text{boundary}} = \sum_{b\in\mathcal{B}} \min_{p_i\in\mathcal{P}} \mathrm{D}(p_ip_{i+1}, b) \tag{8}$$

$$\mathcal{L}_{\text{orthogonal}} = \sum_{p_i\in\mathcal{P}} |\overrightarrow{p_{i-1}p_i} \cdot \overrightarrow{p_ip_{i+1}}| \tag{9}$$

$$\mathcal{L}_{\text{IOU}} = \text{IOU}(\text{Rasterized}(\mathcal{P}), \mathcal{C}) \tag{10}$$

$$\min_{p_i\in\mathcal{P}} \lambda_1\mathcal{L}_{\text{boundary}} + \lambda_2\mathcal{L}_{\text{orthogonal}} - \lambda_3\mathcal{L}_{\text{IOU}} \tag{11}$$

where $\mathrm{D}(p_ip_{i+1}, b)$ represents the shortest distance from a point $b$ to a line segment $p_ip_{i+1}$. Rasterized$(\mathcal{P})$ indicates the area covered by polygon rasterization. It can be implemented by differentiable rendering [6, 23]. IOU means Intersection over Union operation. $\mathcal{L}_{\text{boundary}}$ measures the matching degree between the optimized polygon and the room contour. $\mathcal{L}_{\text{IOU}}$ measures the matching degree between the polygon area and the interior area of the room. $\mathcal{L}_{\text{orthogonal}}$ measures the orthogonality between polygon edges. The weights $\lambda_1, \lambda_2, \lambda_3$ control the importance of each term and are experimentally set to 1,5,1.

- *Room contour vertex number reduction:* When the following two conditions are met, we will reduce the vertices of the polygon.

  - *condition a*: $||p_i - p_{i+1}||_2 < \delta_a$
    When the distance between two vertices of the polygon is smaller than $\delta_a$, the two vertices are merged into one vertex. The coordinate of the new vertex are average of the coordinates of the previous two vertices.

- *condition b*: $|\cos(\overrightarrow{p_{i-1}p_i}, \overrightarrow{p_i p_{i+1}})| > \delta_b$
  When three adjacent vertices of the polygon are approximately collinear, the middle vertex is deleted.

We set $\delta_a$ to half of the wall width, and $\delta_b$ to $\cos(10°)$

**Center line optimization.** Generally, two methods are used to represent the walls in vectorized floor plans. One is to use the edge line to draw the wall, applied in previously optimized room contour polygon. The other is to use the center line to draw the wall. In order to facilitate 3D reconstruction, we introduce how to optimize the center line to draw the wall. A method, similar to the previous room contour optimization, is utilized.

We use $\mathcal{P}_k$ to represent the $k$-th polygon previously optimized, with $K$ polygons in all. The $k$-th polygon has $N^k$ vertexes, represented as $p^k_1, ..., p^k_i, ..., p^k_{N^k}$. Our goal is to optimize a graph $\mathcal{G}$ with vertices $\mathcal{V} = \{v_i | i = 1, .., |\mathcal{V}|\}$ and edges $\mathcal{E} = \{e_i | i = 1, ..., |\mathcal{E}|\}$. The vertices of the graph represent the junctions of the walls, and the edges represent the center lines of walls. Initially, this graph $\mathcal{G}$ is a set of $\{\mathcal{P}_k | k = 1, .., K\}$.

- *Wall junction coordinate optimization:*

$$\mathcal{L}_{\text{center}} = \sum_{a \in \mathcal{A}} \min_{e_i \in \mathcal{E}} \mathrm{D}(e_i, a) \tag{12}$$

$$\mathcal{L}_{\text{nearby}} = \sum_{k}^{K} \sum_{i}^{N^k} ||p^k_i - v_{L(p^k_i)}||^2_2 \tag{13}$$

$$\mathcal{L}_{\text{alignment}} = \sum_{e_i \in \mathcal{E}} \min(|\overrightarrow{e_i} \cdot \overrightarrow{[0,1]}|, |\overrightarrow{e_i} \cdot \overrightarrow{[1,0]}|) \tag{14}$$

$$\min_{\mathcal{V},\mathcal{E}} w_1 \mathcal{L}_{\text{center}} + w_2 \mathcal{L}_{\text{nearby}} + w_3 \mathcal{L}_{\text{alignment}} \tag{15}$$

where $L(p^k_i)$ is a mapping index that maps an initial point $p^k_i$ corresponding to the current optimization point $v_{L(p^k_i)}$. $\overrightarrow{[0,1]}$,$\overrightarrow{[1,0]}$ are the horizontal and vertical direction vectors respectively. $\mathcal{L}_{\text{center}}$ measures how well the generated wall matches the original wall pixels $\mathcal{A}$. $\mathcal{L}_{\text{nearby}}$ term constrains $v$ around the vertices of the initial polygon. $\mathcal{L}_{\text{alignment}}$ restricts the edges we get as horizontal or vertical as possible. The weights $w_1$,$w_2$ and $w_3$ are set to 1.0,2.0 and 1.0, respectively.

- *Wall junction number reduction:*
  - *condition x*: $||v_i - v_j||_2 < \delta_x$
    When the distance between two vertices $v_i, v_j (i \neq j)$ of the graph is smaller than $\delta_x$, the two vertices merge into one vertex. The new coordinates are the average of the coordinates of the two vertices.
  - *condition y*: $\mathrm{D}(e_i, v_j) < \delta_y$
    The two vertices of $i$-th edge $e_i$ are $v^{\mathrm{s}}_{e_i}$ and $v^{\mathrm{e}}_{e_i}$. When the distance between a vertex $v_j$ in the graph and the edge $e_i$ is less than a certain threshold $\delta_y$, we add this vertex to the edge, and this edge $(v^{\mathrm{s}}_{e_i}, v^{\mathrm{e}}_{e_i})$ becomes two edges $(v^{\mathrm{s}}_{e_i}, v_j)$ , $(v_j, v^{\mathrm{e}}_{e_i})$

We set both $\delta_x$ and $\delta_y$ to be half of the wall width.

**Room type refine.** For semantic segmentation, we can obtain the room type distribution by counting the pixel types inside the room. In addition, both symbol detection and text detection provide us with information of the room type. More specifically, for a certain room, if the center of the symbol or text box is within the optimized contour polygon, we vote for the corresponding room type of the box. Then we normalize to get the distribution of the corresponding room. If no box center is in the optimized polygon, it will be treated as a discrete uniform distribution. Finally, we combine these three distributions to select the most likely room type.

**Opening endpoints extraction.** We obtain endpoint candidates by performing non-maximum suppression on the door, window, and doorway heatmaps. After obtaining the location of the endpoint, we obtain the corresponding relationship of the endpoint through the semantic segmentation result. Since the thickness of the wall in the floor plan is different, we need to calculate the thickness of each part of the wall. The detailed calculation method can be found in the supplementary material.

## 4.6. Reconstruction post-processing

The goal of post-processing is to convert the 3D vectorized result into a 3D reconstruction result of real physical size through scaling. If the scale of the residential floor plan is not available, the median length of the door is 0.9m by default, which can be used to calculate scale information. In the 3D reconstruction result, the default height of the wall is 3.2m. Since the real physical size information of the house has been obtained, the type of doors and windows can be determined according to the length of the doors and windows. In our implementation, doors with a length of less than 1.5m are ordinary single doors, those with a length less than 2m are single swing doors, and those greater than 2m are double sliding doors. A window less than 0.8m in length is a small window, less than 1.6m is a ordinary window, less than 2.4m is a large window, and greater than 2.4m is a floor-based window.

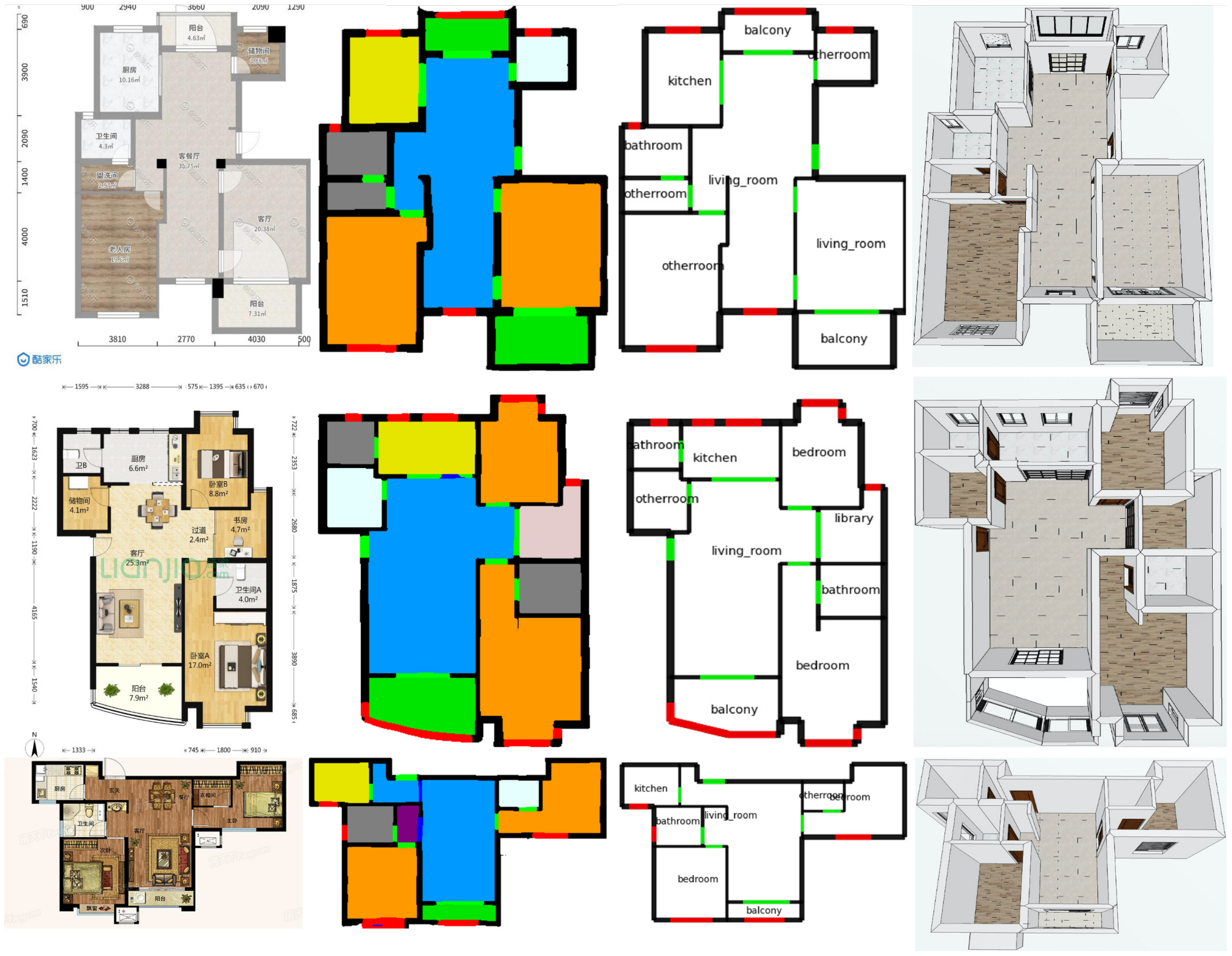

Figure 3. Floor plan recognition and reconstruction results. From left to right, an input floor plan image, semantic segmentation result with post-processing, reconstructed vector-graphics representation with room type annotation, the corresponding 3D reconstruction model. The original images come from Kujiale [4] (top), Lianjia [5] (middle) and Fangtianxia [2] (bottom).

| | | R2V | R2V$_D$ | DFPR | DFPR$_D$ | Ours | Ours$_+$ |
|---|---|---|---|---|---|---|---|
| *Acc.* | | 0.83 | 0.91 | 0.82 | 0.94 | 0.96 | **0.97** |
| *Class Acc.* | Wall | 0.85 | 0.90 | 0.83 | 0.89 | 0.90 | **0.92** |
| | Door | 0.79 | 0.83 | 0.80 | 0.83 | 0.82 | **0.87** |
| | Window | 0.81 | 0.86 | 0.79 | 0.88 | 0.87 | **0.90** |
| | Doorway | 0.55 | 0.67 | 0.57 | 0.65 | 0.67 | **0.79** |
| | Living room | 0.85 | 0.92 | 0.81 | **0.97** | 0.96 | 0.96 |
| | Bedroom | 0.82 | 0.95 | 0.80 | 0.94 | 0.97 | **0.97** |
| | Kitchen | 0.77 | 0.94 | 0.85 | 0.95 | **0.96** | 0.95 |
| | Bathroom | 0.82 | 0.92 | 0.81 | 0.94 | 0.94 | **0.95** |
| | Library | 0.80 | 0.88 | 0.79 | **0.90** | 0.89 | 0.89 |
| | Balcony | 0.86 | 0.97 | 0.82 | 0.95 | 0.96 | **0.96** |
| | Other | 0.72 | 0.87 | 0.74 | 0.83 | 0.85 | **0.88** |
| *mIoU* | | 0.76 | 0.80 | 0.76 | 0.82 | 0.84 | **0.85** |
| *fwIoU* | | 0.82 | 0.90 | 0.83 | 0.93 | 0.95 | **0.96** |

Table 1. Quantitative evaluations compared with R2V [22] and DFPR [33]

## 5. Experiments

### 5.1. Quantitative evaluations

**Comparison with state-of-the-art.** We compare our method with Raster-to-Vector [22] and Deep Floor Plan Recognition [33] (abbreviated as R2V and DFPR), on RFP dataset. Due to reliance on Manhattan's assumption, the R2V method cannot detect inclined walls. To run R2V, inclined walls are deleted when generating the R2V labels, and a heuristics method is utilized to convert our label to junction types. For the DFPR method, pixel categories are divided into room-boundary class and room-type class. For fairness, the results of the R2V and DFPR methods, with ROI detection module, are introduced. We use the subscript D to represent, as shown in Table 1. The results, with (denoted with subscript +) and without(w/o) vectorization, are provided for evaluation on our methods. During the comparison, we used the same post-processing strategy in [33] and followed the rasterization process introduced in [22] to convert the vectorized output into per-pixel.

For quantitative evaluation, the overall pixel accuracy, per class pixel accuracy, mean intersection over union, and frequency weighted intersection over union, abbreviated as *Acc.*, *Class Acc.*, *mIoU* and *fwIoU*, are used as metrics [24, 33]. Table 1 shows the quantitative comparison results. It can be seen that our method can achieve higher results on most floor plan elements, and vectorization can further improve our performance.

**ROI detection.** $AP_{0.5}$ and $AP_{0.75}$ are 0.97 and 0.90 respectively. It can be seen from Table 1 that no matter which method is used, the extraction of the effective area is crucial for recognition.

**Vectorization.** In order to quantitatively evaluate vectorization results, we use the metric in the R2V paper [22]. Compared with other predictions, if the current prediction result is the minimum distance to the target and less than the threshold, the current prediction is correct. For wall junctions, we use the Euclidean distance and the threshold $\tau_w$. For opening primitives, the distance between the prediction and the target is the larger value of the Euclidean distance between the two pairs of corresponding endpoints, and the threshold is $\tau_o$. We set $\tau_w = \tau_o = \text{width}_{\text{wall}}/2$. The results shows in Table 2. It can be seen from the experimental results that our method can produce more accurate results.

In order to evaluate the effectiveness of each constraint in vectorization, we disable one constraint each time and record the performance. Note that we do not evaluate the $\mathcal{L}_{\text{boundary}}$ and $\mathcal{L}_{\text{center}}$ constraints, since they are essential. Table 2 shows that when full modules are performed, precision and recall consistently over the other systems are improved. This reflects that we have obtained an accurate and simplified junction location. The $\mathcal{L}_{\text{orthogonal}}$ constrains the angular relationship between the sides of the polygon, so as to form a simplified polygon. $\mathcal{L}_{\text{nearby}}$ and $\mathcal{L}_{\text{IOU}}$ introduce fine-level constraints to improve wall accuracy. Comparable results were obtained without $\mathcal{L}_{\text{alignment}}$. However, the optimized wall has more inclined walls, which is explained in the supplementary material.

|  | Wall Junction |  | Opening |  |
|---|---|---|---|---|
|  | *Acc.* | *Recall* | *Acc.* | *Recall* |
| R2V$_{\text{D}}$ | 0.95 | 0.91 | 0.93 | 0.92 |
| Ours-$\mathcal{L}_{\text{orthogonal}}$ | 0.67 | 0.71 | **0.97** | 0.75 |
| Ours-$\mathcal{L}_{\text{IOU}}$ | 0.92 | 0.90 | 0.95 | 0.92 |
| Ours-$\mathcal{L}_{\text{nearby}}$ | 0.90 | 0.94 | 0.95 | 0.92 |
| Ours-$\mathcal{L}_{\text{alignment}}$ | 0.95 | **0.95** | **0.97** | **0.93** |
| Ours | **0.96** | 0.94 | **0.97** | **0.93** |

Table 2. Quantitative evaluation of vectorization results

**Others.** Due to space limitations, we have shown other experiments in the supplementary material, such as structural elements extraction, text and symbols detection, scale calculation.

### 5.2. Qualitative evaluations

Figure 3 shows an input floor plan image, semantic segmentation result with post-processing, reconstructed vector-graphics representation with room type annotation, the corresponding 3D reconstruction. The different colors of the segmented and reconstructed vector graphics result represent different elements, and the palette information can be found in Figure 2. We evaluate the generalization ability of the system by processing floor plan images from other data sources. Please refer to the supplementary material for more results and analysis.

### 5.3. Limitation

For an open kitchen, the judgment of the room category will be wrong, because we assume that the room is separated by walls, doors, windows, and doorways. The open kitchen does not meet such requirements. The scale recognition of inclined walls is not supported. Curved walls cannot be vectorized well. In the labeling and recognition of the floor plan, we do not consider the outdoor air-conditioning platform, which is labeled as AC in most cases.

## 6. Conclusion

We present an automatic framework for residential floor plan recognition and reconstruction that accurately recognizes the structure, type, and size of the room, and outputs vectorized 3D reconstruction results. We believe the framework and sub-modules of the system could be utilized as benchmarks and make a significant contribution in fields related to floor plan images.

## References

[1] Autocad. http://www.autodesk.com/products/autocad/overview. 1

[2] Fangtianxia. https://www.fang.com/. 7

[3] Homestyler. https://www.homestyler.com/. 1

[4] Kujiale. https://www.kujiale.com/. 7

[5] Lianjia. https://www.lianjia.com/. 3, 7

[6] Pytorch3d. https://pytorch3d.org/. 5

[7] Sketchup. https://www.sketchup.com/. 1

[8] Sheraz Ahmed, Marcus Liwicki, Markus Weber, and Andreas Dengel. Improved automatic analysis of architectural floor plans. In *2011 International Conference on Document Analysis and Recognition*, pages 864–869. IEEE, 2011. 2

[9] Sheraz Ahmed, Marcus Liwicki, Markus Weber, and Andreas Dengel. Automatic room detection and room labeling from architectural floor plans. In *2012 10th IAPR International Workshop on Document Analysis Systems*, pages 339–343. IEEE, 2012. 2

[10] Sheraz Ahmed, Markus Weber, Marcus Liwicki, and Andreas Dengel. Text/graphics segmentation in architectural floor plans. In *2011 International Conference on Document Analysis and Recognition*, pages 734–738. IEEE, 2011. 2

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 3, 4

[12] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 3

[13] Lluís-Pere de las Heras, Sheraz Ahmed, Marcus Liwicki, Ernest Valveny, and Gemma Sánchez. Statistical segmentation and structural recognition for floor plan interpretation. *International Journal on Document Analysis and Recognition (IJDAR)*, 17(3):221–237, 2014. 2

[14] Samuel Dodge, Jiu Xu, and Björn Stenger. Parsing floor plan images. In *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pages 358–361. IEEE, 2017. 1, 2

[15] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[17] Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, and Juho Kannala. Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis. In *Scandinavian Conference on Image Analysis*, pages 28–40. Springer, 2019. 2

[18] Tsung-Wei Ke, Jyh-Jing Hwang, Ziwei Liu, and Stella X Yu. Adaptive affinity fields for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 587–602, 2018. 3

[19] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015. 5

[20] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 3

[21] Chenxi Liu, Alexander G Schwing, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Rent3d: Floor-plan priors for monocular layout estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3413–3421, 2015. 2

[22] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-vector: Revisiting floorplan transformation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2195–2203, 2017. 1, 2, 3, 7, 8

[23] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019. 5

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 4, 8

[25] Sébastien Macé, Hervé Locteau, Ernest Valveny, and Salvatore Tabbone. A system to detect rooms in architectural floor plan images. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 167–174, 2010. 2

[26] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972. 5

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[29] Ilya Y Surikov, Mikhail A Nakhatovich, Sergey Y Belyaev, and Daniil A Savchuk. Floor plan recognition and vectorization using combination unet, faster-rcnn, statistical component analysis and ramer-douglas-peucker. In *International Conference on Computing Science, Communication and Security*, pages 16–28. Springer, 2020. 2

[30] Xu Tang, Daniel K Du, Zeqiang He, and Jingtuo Liu. Pyramidbox: A context-assisted single shot face detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 797–813, 2018. 4

[31] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *European conference on computer vision*, pages 56–72. Springer, 2016. 2

[32] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of*

*the European conference on computer vision (ECCV)*, pages 466–481, 2018. 4

[33] Zhiliang Zeng, Xianzhi Li, Ying Kin Yu, and Chi-Wing Fu. Deep floor plan recognition using a multi-task network with room-boundary-guided attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9096–9104, 2019. 2, 3, 5, 7, 8

[34] Yuli Zhang, Yeyang He, Shaowen Zhu, and Xinhan Di. The direction-aware, learnable, additive kernels and the adversarial network for deep floor plan recognition. *arXiv preprint arXiv:2001.11194*, 2020. 2, 3

[35] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017. 2